LSTM – Music Generation(Seq2Seq)

Data Preparation

The dataset I used for this project is just to download various midi format songs for various genres which includes –

1. Classical.
2. Jazz.
3. Rock.
4. Pop.

Processing Midi files

For this project I have used midi_parser package in python to process the midi files.

Midi files generally contains multiple tracks, and each track comprises notes. Each track will usually be a specific instrument, and the notes within it would contain details on the pitch, velocity and start/end times of the notes played with that instrument.

We know each track has a list of notes. And each note has start time, end time, pitch, and velocity.

We need to encode these 4 features as a sequence of vectors (matrices) to input this information into our model. The most ideal method of encoding them is to treat this as a multi-variate series and design a many-to-many model, wherein the model would take a series of 4 features as input and give 4 values as output.

So, we need to take 4 variables (start time, end time, pitch, velocity) and somehow mash them up into one value.

1. We find the difference of end time and start time of a note and make a new variable duration.

Duration = end time of note - start time of note

2. We look at the distribution of the values of duration in all the midi files, and split them into 4 bins.

3. We split velocity of the notes into bins

4. Now we concatenate the pitch, bin number of velocity and bin number of durations to form categorical variables.

5. We then one-hot encode the pseudo-notes

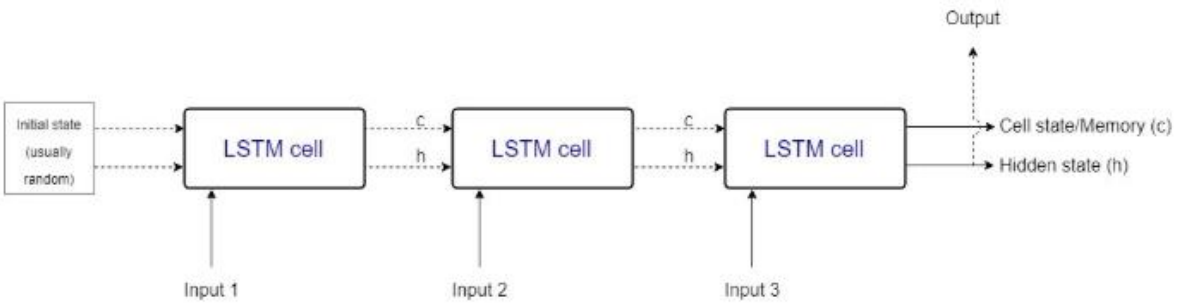
Encoder – Decoder

The sequence to sequence model is an encoder followed by a decoder. The encoders and decoders are essentially LSTMs. LSTMs (Long Short-Term Memory cells) are special kind of Recurrent Neural Networks that can handle long term dependencies.

The encoder takes an encoded sequence of data, and outputs a vector. This vector is then decoded into another sequence by the decoder. The decoder and encoder comprise of LSTM cells.

An LSTM cell takes 2 inputs and gives 1 to 2 outputs depending on how it's configured. LSTMs are generally connected to each other, i.e. outputs on one LSTM is fed into another LSTM.

LSTM cell outputs a cell state and hidden states. The hidden state is the output of the cell. The cell state is a compressed representation of the memory of the neural network upto that point. This memory gives the neural network the capacity to capture long term dependency. Both the states are usually passed onto the next cell.

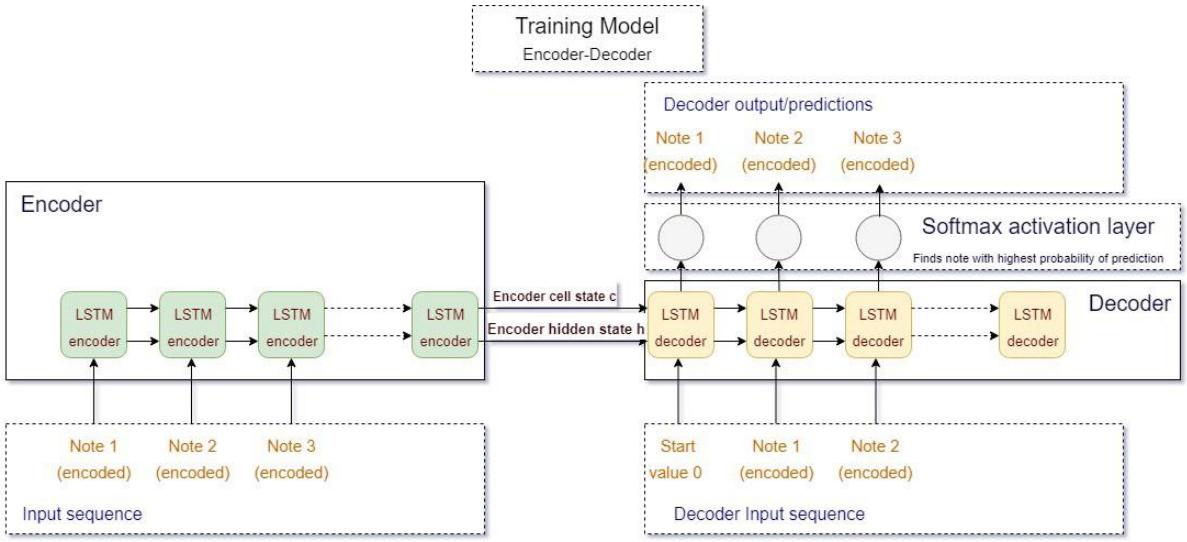


Model

Training Model

The encoder encodes the sequence of input vectors (encoded sequence of notes of the song). The outputs (hidden and cell states) of the encoder is then fed into the decoder. The decoder takes 3 inputs: 2 states from encoder and encoded notes of the song offset by one timestep from the encoder input.

The decoder output is then fed into a softmax activation function and compared to the target data.



Model Inference –

We will be using the above model for generating the sequence. In the training model we are feeding an offset of the desired output as the decoder input. We need to reconfigure the decoder model to generate one output and then feed that output into the next decoder cell to generate the next output and so on.

1. **Input Preparation:**

An LSTM only takes in a 3D vector. Currently, our input is 2 dimensional. Each song is an ordered list of pseudo-notes.

2. **Building Training Model:**

Encoder: 2 LSTM layers having 1024 neurons each with a dropout layer in between them.
The dropout layer randomly loses a portion of the data passing through it. This helps the model regularize and generalize better and reduce the likelihood of overfitting.

Decoder: 2 LSTM layers with 1024 neurons each, followed by a dense layer with softmax activation. The softmax generates a probability distribution for different possible notes as output and we can select the max probability as the predicted output.

3. Building the inference model:

The inference model's encoder is same as that of training model.
For the decoder, the same layers from the training model's decoder is used.

4. Decoding and generating MIDI file:

The output from the above code is a series of pseudo-notes. We then do some quick processing to split this back into start time, end time, pitch and velocity.